COMPUTER
EDUCATION
UNIT

REPRINT: 1985

PROGRAMMABLE TURTLE

Division of Services
NSW Department of Education

CEU 013

FOREWORD

This Curriculum Ideas is the second of two produced as a result of a curriculum workshop on 'Computer Education in Years 9/10'. It contains the unit on the 'Programmable Turtle' which should prove to be of considerable interest and use to teachers in developing Computer Education courses. The units of work are in no way prescriptive.

The document has been reprinted in 1985 to satisfy the continuing need for such material in schools.

Any comments and general enquiries should be directed to:

Chief Education Officer,
Computer Education Unit,
P.O. Box 16,
ERSKINEVILLE 2043

Phone: 519 4599

PREAMBLE

In April 1981, experienced teachers in the field of Computer Education attended a curriculum workshop at Blackheath. At this workshop, groups produced materials which could be used in developing a Computer Awareness course.

A previous Curriculum Ideas, "Support Materials for Computer Education", dealt with ;

A. History and the Computer.
B. Components of a Computer.
C. Uses of Computers.
D. Implications of a Changing Technology.
E. Computer Programming - BASIC .

This issue of Curriculum Ideas consists of notes and exercises on a language called 'Programmable Turtle'.

It is hoped that 'Programmable Turtle' will be of use in the teaching of 'Computer Awareness' by being a simple and an easily operated language.

It should be available on disk from the workshop participants.

Appreciation is expressed to all teachers involved for their willing effort.

Richard Wiktorowicz
Curriculum Consultant (Computer Education)
Directorate of Studies

The Workshop Participants

Bob Baker,	Ryde High School.
Chris Bakon,	Nyngan High School.
Grant Beard,	Maroubra Bay High School.
Bob Brett,	Canterbury Boys High School.
Ted Duffy,	Merrylands High School.
Warren Dunne,	Forest High School.
Kevin Ford,	Mathematics Consultant, North Sydney.
Steve Holbrow,	Shalvey High School.
Karl Holmes,	Barraba Central School.
Bob Hugget,	Richmond River High School.
Carl Jackson,	Figtree High School.
Paul Jenner,	Casula High School.
Graham Kennedy,	Farrer Agricultural High School.
Des Kilroy,	Jannali Boys High School.
David Lofts,	Murwillumbah High School.
Kevin Maddox,	Dungog High School.
Bruce McAndrew,	Crows Nest High School.
John Messing,	Wagga Wagga High School.
John Morgan,	Nyngan High School.
Paul Mulhearn,	Warners Bay High School.
Ian Olney,	Wollongong High School.
Colin Taylor,	Hilston Central School.
Ian Walsh,	Elderslie High School.

UNIT 1.

History and the Computer

Compiled by :

Kevin Ford,
Steve Holbrow,
Des Kilroy.

UNIT 2.

Components of a Computer

Compiled by :

Grant Beard,
Ted Duffy.

UNIT 3.

Uses of Computers

Compiled by :

Bob Brett,
Graham Kennedy,
Ian Olney,
Ian Walsh.

UNIT 4.

Implications of a Changing Technology

Compiled by :

Warren Dunne,
Carl Jackson,
David Lofts,
Kevin Maddox,
John Morgan.

UNIT 5.

Introductory Concepts and Basic

Compiled by :

Bob Baker,
Karl Holmes,
Bob Hugget,
John Messing,
Paul Mulhearn,
Colin Taylor.

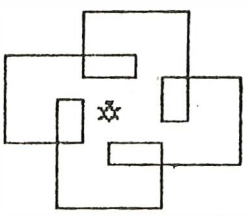
UNIT 6.

Programmable Turtle

Compiled by :

Chris Bakon,
Paul Jenner,
Bruce McAndrew,
John Messing.

PROGRAMMABLE TURTLE

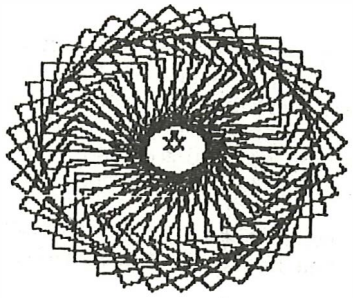


A PROGRAMMING LANGUAGE FOR CHILDREN

PROGRAMMABLE TURTLE

A PROGRAMMING LANGUAGE FOR CHILDREN

CONTENTS



INTRODUCTION

Computer awareness, computer literacy and "hands on experience" are very popular phrases these days. Many schools are conducting computer awareness courses of varying lengths in junior forms and attempting to give the students some sort of hands on experience in teaching the concept of computer programming. The problem is that typing in and debugging a BASIC computer program is such a lengthy process, that it is almost impossible for an entire junior class to have a turn at the computer within a reasonable time.

Even for the students who do manage to type in and run their program, the results are often not terribly exciting. Most students are not very impressed to find that they can get the computer to do Pythagoras' theorem, or add up the numbers from 1 to 10.

Turtle is a programming language for the APPLE II computer that attempts to overcome these problems. It is designed to be used in a classroom situation. Students write computer programs that contain instructions to move a model of a robot turtle around the T.V. screen of the computer. The turtle leaves a trace as it moves so that the output of the program is some sort of graphic figure on the screen.

The instructions are simple, easily mastered in a few minutes, and children are motivated to obtain results. In fact the instructions are so simple that the problems that children have are more likely to be problems of geometry rather than computer programming problems.

It is not necessary for teachers to have any experience with computers or any other computer language to use the Turtle.

SYSTEM REQUIREMENTS

Turtle requires a 48k APPLE II computer with a disk drive. A printer is not needed in classroom use, but can be useful in conjunction with the TEACHER program.

GETTING STARTED

To start the Turtle boot the diskette. See the beginning of the tutorial chapter if you do not know how to do this.

Then you must create a file in your name using the TEACHER program. Having created your file (or files) follow through the tutorial chapter.

HISTORY

Turtle is based on the work of Seymour Papert from M.I.T. Since the 1960's Papert and his associates have been doing a lot of work involving small children using computers. Their work has been based on a computer language called LOGO, of which Turtle Geometry is a part. It is important to realise that this version of Turtle is not an implementation of LOGO, but a limited method of doing turtle geometry for a specific purpose.

LOGO Turtle often uses a real robot turtle than can move on around a large sheet of paper on the floor, and has been successfully used with pre-school children.

This version of Turtle began with John Messing of Wagga Wagga High School. Realising the potential of Turtle for the classroom Mr.Messing wrote a Turtle interpreter for the APPLE utilising single letter commands.

Mr. Messing's program was extended and modified by a team consisting of Chris Bakon, Bruce McAndrew and Paul Jenner at a Directorate of Studies conference held in April 1981. These three are responsible for all faults.

PHILOSOPHY

Turtle is based on the philosophy that in a short introduction to computer programming the following important concepts can be developed;

* That a computer program is a list of instructions that is stored in the computer.
* That somebody (the programmer) has to write this program and enter it into the computer.
* That the program is written in a special language, the rules of which must be followed exactly.
* That the computer will follow this list of instructions implicitly, even if they specify operations different from those that were wanted.
* That the program is often wrong (has bugs) initially and has to be corrected (debugged).
* That the program may be stored temporarily (in the computer's memory) or permanently (on something like a magnetic diskette).
* And that none of these concepts can be properly appreciated unless all pupils have as much opportunity as possible to try things out for themselves on a real computer.(1)

Turtle also allows for slightly longer courses to include the following properties of computer languages.

* That the program may be written so that parts of it are automatically repeated.
* That a lengthy computer program consists of a series of smaller programs, and that any problem is best solved by breaking it into parts.

---

(1) Despite this assertion it should be pointed out that where no computer is available the concepts of turtle programming can still be effectively used as a paper exercise, whether to do with computers or just geometry.

PROGRAMMABLE TURTLE FEATURES

i. PERSONAL SERVICE

Before being able to program the turtle a pupil must sign-on with their name. This is done so that the computer can maintain a file for each pupil containing their personal list of programs. Pupils can only use or modify programs that they themselves have entered. Because of this feature teachers must set up the pupil files using the TEACHER program, before taking the computer to the classroom.

ii. SIMPLE, SINGLE LETTER COMMANDS

To help solve the problem of a pupil's lack of familiarity with the keyboard all commands and instructions consist of one letter, often followed by a number. Also, programs are short and are entered as a single line.

iii. SIMPLE CONCEPTS

Pupils have no trouble understanding Turtle instructions. The problems they do have are of a geometrical nature, and they are of such a concrete nature that they can often be solved by having pupils walk around pretending to be turtles.

iv. IMMEDIATE SYNTAX CHECKING

In programming a computer two sorts of error can be made.

* The first type, called syntax errors, involve errors in the use of the programming language. In other words the computer does not understand the program.

* In the second type of error, often called a logical error, the language is correct, the computer understands but still does the wrong operation. Of course the computer does exactly as it is told and it is the programmers' mistake that causes the error.

Pupils find syntax errors frustrating because they see little in the way of a result for their efforts. Turtle takes care of this by checking each character as it is typed in to make sure that it is in syntax. It will not accept characters which break the rules of the language. Actually the Turtle instructions are so simple that pupils do not make very many syntax errors. The syntax checking feature also drastically reduces the amount of personal attention that the teacher has to give to the pupil at the computer.

Logical errors are less frustrating because the pupil can usually see the mistake from the path that the turtle takes (often wildly different from the intended path).

v. VISUAL PROGRAM TRACE

Pupils programs are printed at the bottom of the screen and the instruction being executed is highlighted.

vi. LOOPS AND SUBROUTINES

Simple methods are provided for repeating sections of a program, and for calling other programs by name from within other programs.

vii. EDITING

Pupils can examine and modify programs that they have stored.

FEATURES NOT PROVIDED.

In the interests of keeping things simple no provision has been made for conditional instructions (the equivalent of IF statements) or variables.

LOGO

Since Turtle is based on the work of Seymour Papert and his associates using the LOGO language, it is appropriate to briefly describe LOGO.

It must be stressed that Turtle is not an implementation of LOGO. LOGO is a complete and very powerful computer language, which is capable of much more than just "turtle talk". Programming in LOGO consists of teaching the computer new words. For example a turtle program in LOGO to draw a square might be;

```
TO SQUARE
FORWARD 20
RIGHT 90
FORWARD 20
RIGHT 90
FORWARD 20
RIGHT 90
FORWARD 20
RIGHT 90
END
```

After defining this program SQUARE using the TO command; SQUARE is a word in LOGO that can be used just like any other word such as FORWARD. In other words, just as in this version of Turtle, users can end up with their own personal version of LOGO.

LOGO is recursive in a much more powerful way than Turtle. For example here is a simple Turtle program in LOGO to draw a circle.

```
TO CIRCLE
FORWARD 1
RIGHT 1
CIRCLE
END
```

The CIRCLE calls itself over and over again in an infinite loop. Papert says that children find this idea of self reference the most exciting of all the ideas that he has introduced to them using computers. The idea is similar to that involved in the old "everything I say is a lie" paradox.

Here is another example of a recursive (non-turtle) program in LOGO.

```
TO FACTORIAL NUMBER
IF NUMBER = 1 THEN FACTORIAL NUMBER = 1
FACTORIAL NUMBER = NUMBER * FACTORIAL NUMBER -1
END
```

LOGO is a list processing language (for the computer buffs it is similar to LISP). The most powerful consequence of that rather simple statement is that LOGO programs can be written to make changes to other LOGO programs.

Papert and his associates see LOGO as being more than just a computer language. They see it as an educational philosophy and as a set of learning tools. They are not enamoured with the usual Skinnerian, behaviourist idea of computers used in education for drill and practice, what they call "the computer programming the child". They see the computer as a learning tool,

providing the opportunity for the child to explore the environment, in situations where "the child is programming the computer".

Using the turtle in LOGO, children learn to be literate in using mathematical concepts in a natural way that connects numbers with their own personal space geometry that they have experienced in learning to move around, without bumping into walls or chairs.

Children explore the language of numbers. Young children using turtle have not been taught about right angles, so they have to experiment to find out that turns of 90 make the turtle move in a "square way". Programming other turtle like objects in LOGO, called sprites, children learn about velocity. They discover that a higher velocity number makes the sprite move faster and are then often astounded when they make the exciting discovery that a velocity of zero makes the sprite stop.

It is obvious that to be comfortable in what Papert calls "Mathland" one must speak the language, just as it helps one to survive in France if one speaks French. The LOGO workers believe that computers provide an environment where children can learn the language of Mathematics just as naturally as they would learn English, French or whatever is the spoken language of their environment.

REFERENCE

Seymour Papert. MINDSTORMS - Computers, Children and Powerful Ideas
HARVESTER PRESS LTD. 1980

THE PICTURES IN THE MANUAL

On the front cover;

```
WORM5-->(F10R90F20R90F30R90F40R90F50R90)4
```

Before the Introduction;

```
SQCIRCLE-->((F50R90)4R90F2L80)36UB12
```

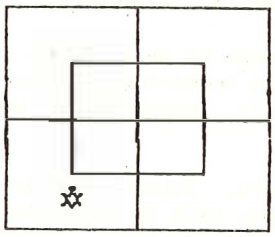
Top of Chapter One;

```
SQ-->(F50R90)4
SQUARE5-->(#SQ#R90)4UL90F25R90B25D#SQ#UB15
```

The programs for the letters of the word "TURTLE" are given in the Tutorial chapter.

Top of Chapter Four;

```
POLY-->UB75(F140R144)5UF12L90F30R90
```



CHAPTER ONE

THE TEACHER PROGRAM

The TEACHER program is provided to allow the teacher to manage the Turtle files.

The Turtle Programming Language is designed so that individual pupil's programs are stored on the disk under a file name that is the pupil's name or initials or, if you like, some other password.

The programs are stored so that a pupil's work done in typing programs does not have to be redone at each Turtle session and of course to demonstrate to pupils the way in which the computer can store information.

The programs are stored under a different file name for each pupil so that each pupil can only use (and more importantly modify) his or her own programs. When each pupil begins a session with Turtle they sign on with their name. If a file has not been created in their name, Turtle will not recognise them.

Thus a file must be created by the teacher entering the names of all of the pupils in a class before Turtle is used in the classroom. The TEACHER program is provided for this purpose. The TEACHER program also allows the class teacher examine pupil programs and to make copies of the Turtle disk.

In most cases the TEACHER program is easy to use if you read the information on the screen carefully.

To use the TEACHER program, start the Turtle in the normal way by booting the disk. If you don't know how to do this read the first part of the tutorial chapter. When the 'NAME PLEASE' prompt appears type the word 'TEACHER'. The TEACHER program will be loaded from disk and the following title page will appear;

```
PROGRAMMABLE
TURTLE

TEACHER MANAGEMENT


1. ADD/DELETE PUPILS
2. PRINT PUPIL INFO
3. START 'TURTLE'
4. MAKE DISK COPY
5. FINISH



WHICH ONE (1-5):
```

Just type the number of the option that you want.

1. ADD/DELETE PUPILS

This option allows you to enter the names of the pupils who are going to be using the Turtle, just type the names in followed by RETURN.

If you are adding names to a list which already exists, just press RETURN repeatedly to go through the existing names.

If you want to modify a name, backspace to the position you want to change when the name appears.

You can go back a name or delete a name by following the instructions on the screen.

2. PRINT PUPIL INFO

This option allows the teacher to examine pupils' programs. You may make lists of their programs on a printer if you have one attached in slot #1 or view them on the screen. This option also allows you to delete pupil programs if the disk is becoming over-full. Although pupils may modify their programs in any way, there is no way for a pupil to delete one of their own program names once it exists.

3. START 'TURTLE'

This option returns you to the main TURTLE program.

4. MAKE DISK COPY

This option allows you to make a copy of the Turtle onto a new disk. Therefore we have a parthenogenic, self replicating Turtle. This option copies all of the files needed to run the Turtle, but it does not copy any of the pupil files.

```
*****************
*               *
*    WARNING    *
*               *
*****************
```

The copying procedure initialises the copy disk, that is it will wipe out any information that is already on the copy disk.

If you want to make a copy of Turtle using some other process (MUFFIN, DEMUFFIN OR FID) then these are the files that you will need to copy;

TURTLE
LOMEM:
TURTLE.SH
TURTLE GRAPHICS
TEACHER
EX1
EX3

5. FINISH

This option allows you to leave the Turtle suite of programs and enter APPLESOFT. It is the only proper way of leaving Turtle. No way has been provided for pupils to exit Turtle from the main Turtle program. They should of course be taught to sign off from the Turtle using the 'Q' command. After a 'Q' command and at the 'NAME PLEASE!' prompt no harm is done to the Turtle if the computer is turned off.

If you have exited Turtle using option five of the TEACHER program and you want to program in APPLESOFT, or run another APPLESOFT program you should first execute an 'FP' command.

CHAPTER TWO

TURTLE TUTORIAL

WHAT IS THE TURTLE ?

The turtle is a little robot with a pen in its tummy that runs around leaving a trace of where it has been. We can't afford a real robot that runs around the room, so we represent the turtle with a small drawing on the computer's T.V. screen.

You can control the movement of the turtle by giving it instructions using the Turtle Programming Language. You talk to the turtle by typing your instructions using the computer keyboard. The Turtle Language is easy to learn and the computer will help you learn it by not letting you type in most of the sorts of mistakes that you can make.

HOW TO GET THE TURTLE GOING

To start the Turtle you need an APPLE II computer with 48K of memory and the Turtle diskette. The diskette is a small, square black plastic envelope containing a disk of magnetic material. This magnetic material is the same sort of stuff that is used in a cassette tape and is very sensitive. If you have not used a diskette before have someone show you how to care for them.

The first step is to put the diskette into the disk drive. Open the door on the drive, and holding the diskette in your right hand with your thumb on the label, carefully insert the diskette all the way into the drive. Then carefully close the door on the disk drive.

The next steps depend on what sort of APPLE you are using.

If you are using an APPLE II Plus all you have to do is turn the computer on (the switch is on the back).

If you are using an older APPLE then follow these steps;

* turn the computer on
* press the RESET key a couple of times
* type the number 6
* while holding down the CTRL key press the P key
* press the RETURN key.

If you have done the above properly the disk drive will make a whirring sound and the 'in use' light will come on. After a while you will see the turtle introducing itself.

HOW TO INTRODUCE YOURSELF

After the turtle has shown off a bit you will see a title page. At the bottom of the screen the following prompt will appear;

NAME PLEASE

Enter your name and press the RETURN key. The disk drive will start up and one of two things will happen;

* If the TEACHER has told the computer your name it will recognise you and come back with READY prompt. Of course you should make sure that it is really you that the computer recognises and not somebody with the same name.

* If the computer does not recognise you it will tell you to try again. If you have made a typing mistake you can try again. The computer will let you have three goes before making rude noises at you. If the computer does not know your name you should ask the teacher to fix it up for you.

PLEASE FINISH PROPERLY.

Of course you would not want to finish yet but there is only one way to finish properly so this is a good time to tell you how.

ALWAYS finish a session with the Turtle by pressing the 'Q' key.

This way the computer will remember your programs properly for your next session and set itself up for the next person.

HOW TO GET NEW PAPER FOR THE TURTLE

You should be looking at a prompt that says READY with a flashing white square after it.

Press the 'C' key.

This is the CLEAR command. You should see the screen clear and a little picture of the turtle appears in the middle of the screen. The READY prompt will appear at the bottom of the screen. The turtle is facing towards the top of the screen with its pen DOWN.

HOW TO MAKE THE TURTLE MOVE

You can make the turtle move about the screen. Type this;

F20 (followed by the RETURN key).

You will see the turtle move 20 screen spaces FORWARDS leaving a trace as it moves. Now try this;

B30 (followed by RETURN)

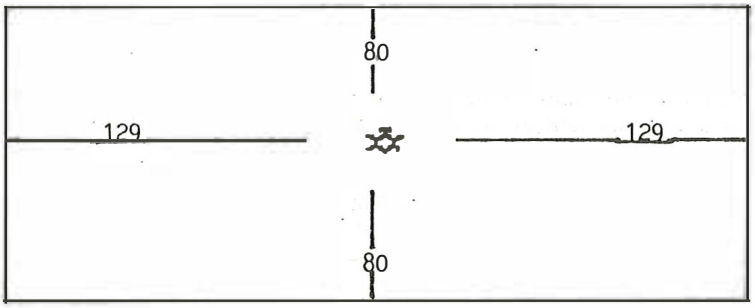
and the turtle will move 30 spaces BACKWARDS.

The letters 'F' and 'B' are INSTRUCTIONS for the turtle meaning FORWARDS and BACKWARDS. The Forwards and Backwards instructions MUST be followed by a number telling the turtle how many spaces to move.

Every set of instructions must be followed by the RETURN key so that the computer knows that you have finished typing. I will not remind you about the RETURN key again.

HOW BIG IS THE SCREEN ?

The screen is 259 spaces wide and 158 spaces high.



Try F300. You will see a message telling you that the turtle would go off the screen.

THE TURTLE TURNS ON THE SPOT.

Having a turtle that can only walk in a straight line is not much use so there are also instructions to make it turn. Try typing the instruction R90.

You should see the turtle turn right through 90 degrees. Notice that the turtle has not moved but only changed direction. The only instructions that make the turtle move are 'F' and 'B'.

You can also make the turtle turn left by using the 'L' instruction.

The 'R' for RIGHT and the 'L' for LEFT instructions MUST be followed by a number telling the turtle how many degrees to turn through.

You can turn the turtle through any number of degrees but the little picture will only be drawn to the nearest 45 degrees.

Now you know how to move the turtle around the screen. You only have to remember the F, B, R, and L instructions. Have a play, see if you can make it draw a square, or a triangle.

Remember you can use the 'C' command any time you want to start over with a clear screen.

YOU CAN CONTROL ITS PEN.

The 'U' instruction puts the turtle's pen UP.

The 'D' instruction puts the pen DOWN.

Using the 'U' instruction you can move the turtle around the screen without leaving a trace. TRY IT!!

Remember that after doing a 'C' command to clear the screen the turtle's pen is ALWAYS DOWN.

STRINGING INSTRUCTIONS TOGETHER.

By now I hope you have worked out a series of instructions that move the turtle in a square. You don't have to use RETURN after each instruction. The

set of instructions can be strung together in one line to make a PROGRAM for the turtle.

For example to draw a square we might use this program;

READY —>F20R90F20R90F20R90F20R90

As the turtle draws the shape you can see your program at the bottom of the screen. The actual instruction that the turtle is doing is highlighted.

Notice that the last R90 in the example is not really necessary to draw the square. It's nice to have it there though because it means that the turtle finishes in the same place that it started AND pointing in the same direction. This can be useful later on.

IT'S HARD TO MAKE MISTAKES.

If you haven't already found out what happens, try typing some wrong letters. If, for example, you type 'X', which does not mean anything to the turtle, you will hear a beep, and the flashing white square will not move on. This checking is called SYNTAX CHECKING.

IT EVEN HAS BRAKES.

Put the example program for a square in again and while the turtle is working press the ESC key. The turtle will stop and a message will appear at the bottom of the screen giving you a choice of pressing the ESC key again to stop altogether, or pressing any other key for the turtle to continue.

THE TURTLE CAN REMEMBER.

By now you have probably typed the program for a square two or more times. You might be thinking that a useful computer should be able to store its programs once they have been typed in.

This is easy to do. So far you have been programming the turtle in IMMEDIATE MODE. That is the turtle does what you tell it to do straight away. We will now look at the PROGRAM MODE.

To tell the computer to store a program type the command 'P'. (Notice that like the 'C' command a RETURN is not wanted here.)

You will then see a message which reads;

```
ENTER
  WHAT IS THE NAME OF YOUR PROGRAM?
NAME -->
```

You now have to give your program a name. Let's give our square program the rather sensible name of 'SQUARE'. Type it in...

Now you will see a prompt which is your program name;

SQUARE-->

Type in the program;

SQUARE-->F20R90F20R90F20R90F20R90

When you have pressed RETURN after typing in the program you will see the READY prompt again and a sign telling you to do a 'C' command. DO IT!

Alright so now you see the turtle in the middle of the screen but it is not drawing a square. All we have done in program mode is told the computer to store the program, we still need to do another command to actually make the turtle draw the shape.

Program names can be made up of any characters but MUST NOT have spaces in them, i.e. you could NOT have a program called TWO SQUARES.

YOUR OWN SHAPE LIST.

Before we tell the turtle to draw our shape let's look at the 'S' for 'SHAPE LIST' command. Type an 'S' (no RETURN needed).

You will then see a list of turtle programs that the computer is storing for YOU. There will only be one at this stage i.e. SQUARE. You will not see programs that are stored for anybody else and similarly they will not be able to get at programs that belong to you.

Remember that you must use the 'Q'command to finish of a session with the turtle. This makes sure that the computer remembers any programs you have put in your shape list for the next time you use the turtle.

Do a 'C' command to clear the screen when you have finished looking at your shape list.

THE DRAW COMMAND.

The command to tell the turtle to do a program you have stored is '#'.(To type a '#' you need to hold down the SHIFT key while typing '#' Like other commands no return is needed.)

After typing '#' you will see;

```
DRAW
  WHAT IS THE NAME OF YOUR PROGRAM?

NAME -->
```

Type in the name of the program that you want done (e.g. SQUARE), and watch in delight.

WHAT IF WE MADE A MISTAKE?

Your program might not do exactly what you want it to. In this case you can make changes by using EDIT MODE.

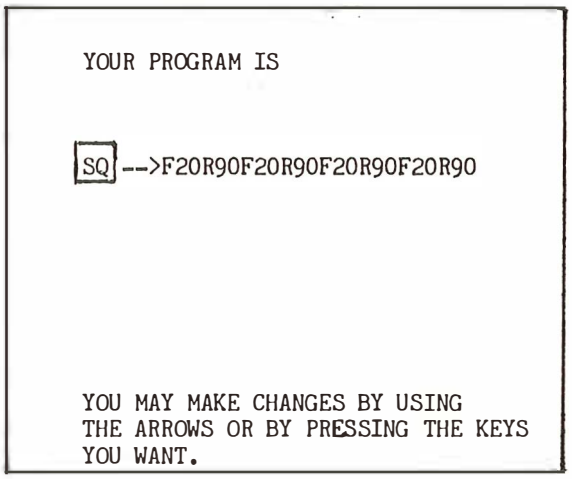
To enter edit mode use the command 'E'. You will then be asked for the name of the program you want to edit.

```
EDIT
  WHAT IS THE NAME OF YOUR PROGRAM?

NAME -->
```

After you have typed in the program name you will see your program printed on the screen, with the flashing white square at the beginning of the program.

You should now see the following;



You can copy over bits of the program that you do not want to change using the forward arrow key ->.

You can back up using the back arrow key <-.

You can change each character of your program by typing the correct character when the flashing white square is over the character you want to change.

If you want to insert a character you can make room for it by typing ctrl-I, when the flashing white square is over the place where you want to insert. The ctrl means that you hold down the CTRL key while you type 'I'. If you want to make room for more characters type ctrl-I again.

You can delete characters from your program by typing ctrl-D when the flashing white square is over the character you want to delete. Ctrl-D also will delete spaces created by ctrl-I.

Notice that even if you have made all the changes you want to make you still have to move the flashing white square right to the end of your program before you press RETURN.

The arrows, ctrl-I and ctrl-D can be used in program mode as well as in edit mode.

EXPERIMENT! Change the SQUARE program so that the turtle draws a square of side 30. Change it so that the turtle moves back 20 spaces before drawing the square, i.e. to

SQUARE-->UB30DF30R90F30R90F30R90F30R90

At any time when you are typing turtle instructions you can cancel the whole line by typing '/'. This can be used in immediate, program or edit mode.

If you accidentally enter program mode, edit mode or draw mode you can get back to READY by just pressing RETURN.

USING YOUR SHAPE LIST

Once a program is stored in your shape list you can use it in other programs. Make sure you have a program stored called SQUARE that draws a square of side 20, then enter the program called TWOSQUARE.

TWOSQUARE -->UB40D#SQUARE#UF40D#SQUARE

Clear the screen and make the turtle do TWOSQUARE. You will see that when the turtle gets up to the #SQUARE# instruction it will call up the SQUARE program from your shape list and do it.

Use this feature! You should not write long programs to do complicated drawings, but short programs that draw bits of the drawing and a final program that connects the bits together.

EXAMPLE 1

The Turtle's name at the top of this chapter can be drawn with these programs.

T -->UF20R90DF20B10R90F20R90UF10R90

U -->DF20UR90F20R90DF20R90F20R90U

R ->DF20R90F20R90F10R90F10L135F14B14R135
F10L90F10R180U

L -->DF20UR90F20R90F20R90DF20R90U

E -->DF20R90F20UR90F10R90F5DF15L90F10L90
F20B20L90U

If you follow the above programs through you will see that they all finish with the turtle back in its starting place, pointing north.

To spell a word using these programs we will always need the same steps to put the space between the letters, so we need this program as well.

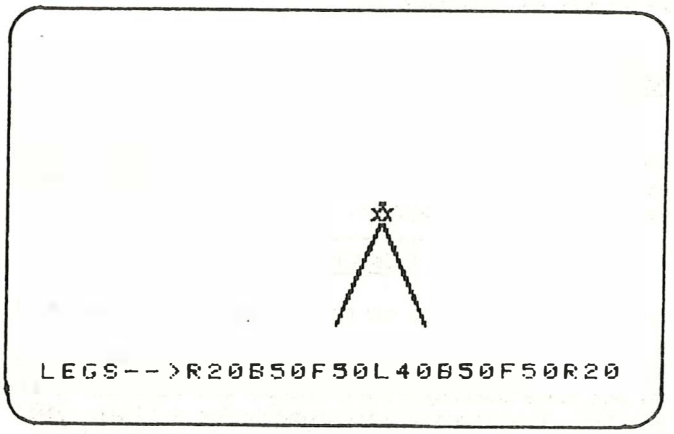
SP -->R90F15L90

Now putting them all together to spell "TURTLE" (moving the turtle to the left first);

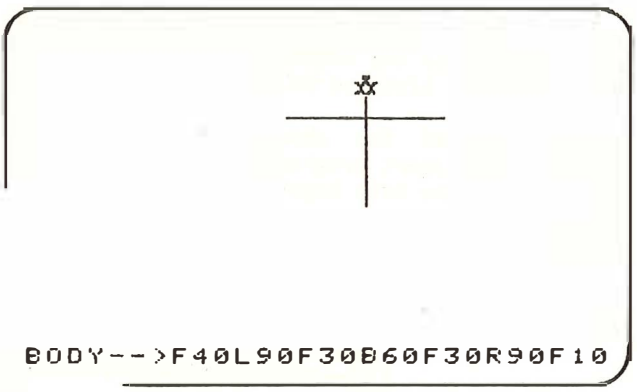
TURTLE -->UL90F20R90D#T##SP#U##SP##R##SP
##T##SP##L##SP##E#F30

EXAMPLE 2

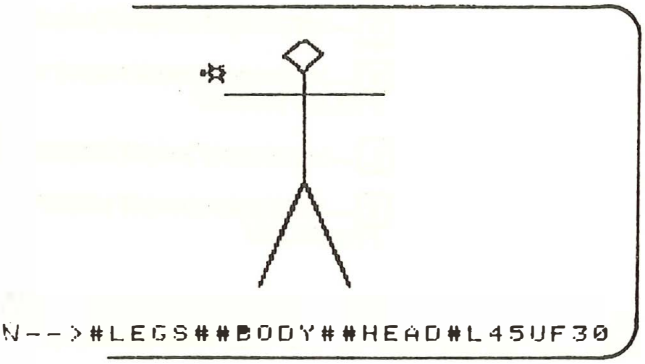
We start with this program to draw some LEGS. Notice that it brings us back to our starting point.



And this program to draw a BODY.



```
BODY-->F40L90F30B60F30R90F10
```



```
PERSON-->#LEGS##BODY##HEAD#L45UF30
```

And put them together with a program called HEAD to draw a PERSON.

PROGRAMS THAT BITE THEIR OWN TAILS.

We can get a program to call itself. For example here is another program to draw a square.

```
SQ -->F20R90#SQ#
```

If you try this out you will have a square alright but the READY will not come back. This is because the F20R90 part is being done over and over again. The program will not stop until the computer runs out of memory. You can stop it using the ESC key.

Can you this idea for a program to draw a circle?

In computer talk this sort of thing is called a recursive program. It's a powerful idea although it's not much use in the turtle language because there is no way of stopping it.

REPEATING TURTLE.

As we saw above we can make the turtle to do things over and over again. There is another way of doing this called LOOPING. Here is another version of a program to draw a square.

```
SQLOOP -->(F20R90)4
```

Try it out! The part in parentheses () is repeated four times.

Your program MUST contain matching parentheses i.e. as many ('s as)'s. If you do not have matching parentheses the syntax checker will not let you hit RETURN. The closing parenthesis for a loop MUST be followed by a number telling the turtle how many times to do the loop.

You can do loops inside loops. Computer programmers call this nesting loops.

For example;

THREESQ -->B60((F20R90)4F30)3

or another version of the same thing;

THREESQ -->B60(#SQLOOP#F30)3

Try a program for yourself to draw an equilateral triangle using loops.

THE TOTAL TURTLE TRIP THEOREM.

Did the triangle program work? A lot of people get it wrong the first time because they tell the turtle to turn through the wrong angle.

If the turtle travels all the way around a figure and ends up facing the way it started then it must have turned through a total of 360 degrees. This fact is called the Total Turtle Trip Theorem (TTTT).

We can use the TTTT to write a program to draw a pentagon (a figure with five sides). The turtle must turn through five angles so each angle is 360 divided by 5 which is 72 degrees. Here is the program;

PENTAGON -->(F20R72)5

Try writing a program for a hexagon (six sided figure) and a circle.

THE INVISIBLE TURTLE.

After you have a program that is working you can speed it up by making the turtle invisible. To do this use the 'T' command. After typing 'T' you will be given the option of turning the turtle on or off. You will also be given the option of turning off the program trace that appears at the bottom of the screen when the turtle is drawing a shape.

REMEMBER TO SAY GOOD-BYE.

```
****************************************
*                                      *
*       Don't forget to use the        *
* 'Q' command when you have finished.  *
*                                      *
****************************************
```

HAVE FUN

****

COMMANDS AND INSTRUCTIONS

The turtle is a small robot which can move about when given certain instructions. Rather than spend the money on one of these robots the Turtle language has been modified so that the movements of the turtle show up on the TV screen.

To start with, the Turtle recognises 2 main types of information :

COMMANDS and INSTRUCTIONS

* COMMANDS are instructions to the computer to decide what you want the computer to do. COMMANDS control the operation of the whole computer.

* INSTRUCTIONS are messages to the turtle to make it do something. Instructions while going to the turtle via the computer, do not change the state of action of the computer. This will become a little clearer later !

When a task such as a command or instruction has been finished the computer displays the READY signal.

SUMMARY OF COMMANDS

```
+-----------------------------------------+
! NOTE:     EACH OF THE FOLLOWING COMMANDS !
! IS ONLY EXECUTED IF IT IS THE FIRST KEY  !
! PRESSED AFTER  THE READY SIGNAL HAS BEEN !
! GIVEN                                    !
+-----------------------------------------+
```

COMMAND 1 C (clear)

Typing this command clears the screen so that the path of the turtle can be started again. When CLEAR is used the turtle starts off in the centre of the screen facing forward and with the pen in the down position.

After having cleared the screen , notice how the READY signal appears.

COMMAND 2 P (program)

Rather than just telling the turtle to draw a shape we can actually teach it a shape so that it can draw it later. To do this we use the 'P' command which will prepare the turtle to memorise a list of instructions.

You will first be asked to give the turtle a name to call your program which you type in at the keyboard. By pressing the RETURN key at the end of the name you will then be allowed to enter your program of instructions.

COMMAND 3 # (draw)

Once you have given the turtle a program to remember (using the P command) you may then ask it to draw that program by pressing the '#' key. You will then be asked to give the turtle the name of the program you wish it to draw, which you do by typing it on the keyboard. When you press the RETURN key you will be able to watch the turtle draw your shape in front of you. Note: You must have executed a 'C' command to clear the screen before you ask to draw the shape.

COMMAND 4 S (shape list)

This command will display a list of all the programs that you have previously instructed the turtle to remember. You will need to execute a 'C' command after looking at your list of shapes before you can draw any more shapes.

COMMAND 5 E (edit)

This command allows you to edit programs that you have previously instructed the turtle to remember. After typing this command you will be asked to give the name of the program that you wish to edit (press the RETURN key after you have typed in the name of your shape).

This command will also require you to execute the 'C' command before you will be able to draw any more shapes.

COMMAND 6 Q (quit)

You should execute this command when you have completed your turn at the computer. It will save any new programs that you have made and prepare the computer for the next student.

COMMAND 7 T (turtle switch)

This command enables you to make the turtle invisible and to turn off the program trace, in order to speed up long programs.

COMMAND 8 <ESC> (pause)

If you press the <ESC> key while the turtle is drawing a shape it will pause and wait for you to tell it to go on. Pressing the <ESC> key again will cancel the shape altogether. Pressing any other key will start the turtle again.

SUMMARY OF INSTRUCTIONS

The turtle can recognise 6 major instructions and 3 house keeping instructions. The major instructions are pen up, pen down, right, left, forward and backwards. These 6 instructions are all that are required to make the Turtle move.

INSTRUCTION 1 U (up)

This is the abbreviation used for pen up. With the pen in this position the Turtle will not leave a mark as it moves around the screen. This will continue until a 'D' instruction is encountered.

INSTRUCTION 2 D (down)

This means pen down and any movement shows up as a white line. This will continue until a 'U' instruction is encountered.

INSTRUCTION 3 R (right)

This stands for turn Right and must be followed by an angle through which it must turn. e.g., R90 or R270. When the turtle turns, it turns the angle mentioned and faces in a new direction. Turning is done on the spot and does not show up on the TV screen.

INSTRUCTION 4 L (left)

This means turn Left and the same comments apply as for R. Examples of the use of this would be L90, L180 etc.

INSTRUCTION 5 F (forward)

This stands for Forward. This is one of the two instructions that actually makes the turtle change position. The letter F must be followed by a number telling it how far forward it is to move e.g., F10, F40 etc. Each unit is about 1mm on the TV screen. The turtle moves forward the required number of spaces in the direction it is facing !

Sometimes the instruction to move forward will cause the turtle to go off the screen. In this case the turtle will stop executing its instructions, showing the faulty instruction. Screen size is 250 units across by 150 units down.

INSTRUCTION 6 B (backwards)

This stands for Backwards and the same comments apply as for F.

INSTRUCTION 7 (

This instruction is given if you wish to repeat a series of instructions a certain number of times. The series of instructions to be repeated must be enclosed between this left parenthesis '(' and a right parenthesis ')' - see next instruction.

INSTRUCTION 8)

This instruction is the partner to the previous one. It is used to signal the end of the sequence of instructions which are to be repeated and it must be followed immediately by a number representing the number of times the instructions are to be repeated.

NOTE: You will not be allowed to end a list of instructions unless there are the same number of left and right parentheses - this is a common error so check this carefully.

NOTE: You are allowed to use up to 40 pairs of parentheses at any one time and they may occur one after another or enclosed inside each other.

INSTRUCTION 9 #

This instruction is similar to the '#' command described before, except that it can be used to draw a previously programmed shape from within another program. Having pressed the '#' key you must follow it with the name of the shape that you wish drawn at that point. When you have finished typing that name you must finish with a second '#' instruction before you can continue with further instructions for the turtle. i.e. Whatever is typed between the two '#' signs is interpreted as being the name of a previously programmed shape. This is emphasised by the shape name being displayed in reverse mode on the screen.

SUMMARY OF SPECIAL COMMANDS

THE FOLLOWING SPECIAL COMMANDS MAY BE EXECUTED AT ANY TIME WHILE ENTERING INSTRUCTIONS.

SPECIAL COMMAND 1 / (disregard)

If this command is executed while you are in the middle of giving the turtle a list of instructions, then those instructions will be ignored and the READY signal will be displayed. This command should be therefore be used if you decide that you don't want to continue with the current program.

CAUTION- Avoid accidentally pressing this key or else you may lose all the hard work that you have just entered.

SPECIAL COMMAND 2 --> (right arrow)

When you press this key the flashing square will move one place to the right. It is used to re-read instructions which are to the right of the flashing square (this is most often used when editing a program or inserting or deleting instructions).

SPECIAL COMMAND 3 <-- (left arrow)

You may use this command to move the flashing square one place to the left.

NOTE: If using this command in IMMEDIATE mode it will delete any characters that it passes over. In PROGRAM or EDIT mode it will move the flashing square over each instruction and leave it untouched. (see command 2 to re-read these instructions).

SPECIAL COMMAND 4 CTRL I (insert)

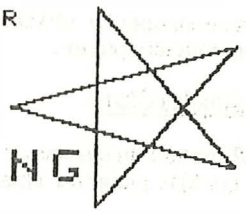
This command is executed by holding the 'CTRL' key down while you press the 'I' key at the same time. This command can be used if you find that you need to insert extra instructions in the middle of the instructions you have already typed.

To do this, use the forward and backward arrows to position the flashing square at the place where you wish to insert your extra instructions. Then press the 'CTRL I' command and all instructions from the flashing square onwards will be moved forward one place leaving a '-' symbol in the space you have made. You may then either add more spaces or begin typing your extra instructions.

SPECIAL COMMAND 5 CTRL D (delete)

This command may be executed by holding the 'CTRL' key down while pressing the 'D' key. This is used to DELETE any instructions that you no longer want.
To do this, use the forward and backward arrows to position the flashing square over the symbol that you wish to delete and then type the 'CTRL D' command. The symbol will disappear and the rest of the instructions will be moved back one place.

CHAPTER FOUR



TEACHING IDEAS

SOME IDEAS FOR CLASSROOM USE.

Of course there are many different ways in which Turtle may be used in the classroom, of which the following are just a few suggestions.

i. BE PREPARED

It is essential that teachers should be thoroughly familiar with the operation of Turtle before using it in the classroom. Every effort has been made to facilitate classroom use but there are minor problems that can only be overcome with familiarity.

ii. DON'T EXPECT TOO MUCH

Remember that Turtle has basically been designed to teach the very fundamental concepts of what a computer program is. Don't expect that all of the pupils of a class will be drawing wonderful designs, and understanding all of the tricks of Turtle in the first few periods. If every member of the class manages to run one program properly, then you will have made good progress.

iii. ALTERNATE EXERCISES

Even though Turtle has been designed for quick turn around of pupils at the computer there will be times when they have finished preparing their program and are waiting for computer time. It is suggested that alternate exercises be provided to occupy pupils during these times. These exercises could be;

* MORE TURTLE - Turtle programming exercises to be corrected in class but not run on the computer.

* PRETEND TURTLE EXERCISES - Turtle programs that are decoded by the pupils. Pupils draw the result of the program in their workbooks.

* COMPUTER AWARENESS WORKSHEETS - If Turtle is being studied as part of a computer awareness course then these exercises may consist of worksheets on other aspects of computers such as history, applications or social impact.

* MATHEMATICS ASSIGNMENTS - Turtle could be made available in periods devoted to revision assignments.

iv. ENCOURAGE WALKING

Pupils who have trouble with the geometrical aspects of Turtle should be encouraged to play Turtle and if necessary act out their programs by walking around the room. This usually provides for some hilarity.

v. ENCOURAGE PROPER PLANNING

Check to see that pupils have properly written out their programs in their workbooks before they go to the computer.

vi. ENCOURAGE USE OF SUB PROGRAMS

If pupils are writing complicated programs, encourage them to split the job into smaller sections. A turtle program should rarely be longer than two lines on the screen.

vii. SIGN UP BOARD

Set aside a section of the blackboard for pupils to sign up in order for computer time as they finish preparing their programs. This can avoid fights. Speaking of blackboards, it is wise to keep the computer, particularly the disk drive, as far as possible away from chalkdust.

viii. INITIALS FOR NAMES

Pupils have a lot of trouble with the typewriter keyboard so it might speed things up a little if initials are entered from the TEACHER program instead of full names.

ix. STUDENT MONITORS

As with everything else some pupils catch on faster than others. You might be surprised which ones do catch on to Turtle quickly. Use these pupils as a resource. Let them help other pupils who are having difficulty. If pupils are helping others at the computer, make sure that they are not the type to butt in too much.

x. OUTSIDE THE CLASSROOM

Make the computer available at lunchtimes and recesses for pupils who become particularly interested.

xi. HARD COPY

If you have a printer that can do graphics dumps, make copies of pupils' work for display in the classroom, or for pasting in workbooks.

A special Turtle command has been incorporated in version 2.0 (or later) of Turtle to do this.

The command is G and is used just like any other command (such as C, P, E etc) when the READY prompt appears. The G Command will not be accepted until it has been enabled. It is enabled by typing the word "DISKETTE" when the NAME PLEASE prompt appears.

After having done this, the G command will cause the message "NAME OF PICTURE" to appear. Type in a name for the picture, (followed by RETURN) and the current picture will be saved on disk. You should not use the Turtle disk, with the pupils' programs for this, because of the room that each picture takes up on the disk.

You can then later reload this picture (and others saved in the same way) and print them out using a graphics dump program.

The picture can be sent directly to an Epson printer by enabling the G command with the word "EPSON" at the NAME PLEASE prompt. The G command will then give you the option of enlarging or inversing the printout and will send the current picture directly to an Epson printer in slot#1.

It should be possible to do the same thing for a Silentype printer with a routine similar to the one at lines 9200,9290 in the TURTLE program.

The G command can be disabled with the word "NOGRAPH" at the NAME PLEASE prompt.

Note that you should be careful to avoid print head overheating problems if you are doing many graphics dumps, especially inversed print outs.

EXERCISES

The sorts of exercises that can be set using Turtle are unlimited. Here are some suggestions.

i. Write programs for the following

   a. The first letter of your first name
   b. A square
   c. A rectangle
   d. A rhombus
   e. A parallelogram
   f. Any triangle
   g. An equilateral triangle
   h. A hexagon

ii. Write programs to;

   a. Draw a dotted line from one side of the screen to the other.

   b. Draw a 10 by 10 square coloured in.

   c. A circle

   d. A small street map. Do not have more than five streets and have T intersections as well as crossings.

   e. Explore worm patterns or spirolaterals. A simple order three worm takes one step, turns right, 2 steps turns right, 3 steps turns right and repeats the whole process ad infinitum. An order five worm pattern is on the front of this manual. See Martin Gardiner's articles in Scientific American and Lorraine Mottershead's book "Spirolaterals".

   f. Fill the screen with squares.

   g. Fill the screen with hexagons.

   h. Draw a picture of any sort. (House with garden, a car etc).

iii. Write programs to draw the letters of the alphabet. Examples are supplied on Worksheet 1.

iv. Draw on graph paper the output of supplied programs. Examples of this are on Worksheet 2.

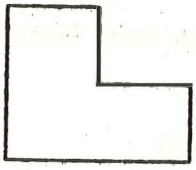
v. Write programs for shapes drawn on graph paper. Examples are on Worksheet 3.

vi. Worksheet 4 consists of a mixture of exercises from Worksheets 3 & 4.

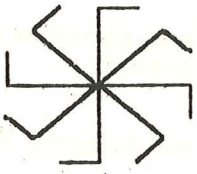
vii. Write programs using loops to draw the following shapes.

a)

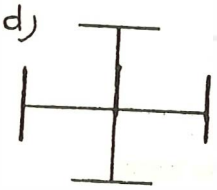
b)



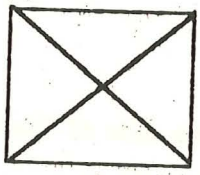
c)



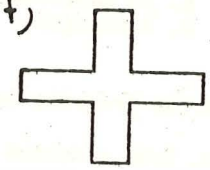
d)



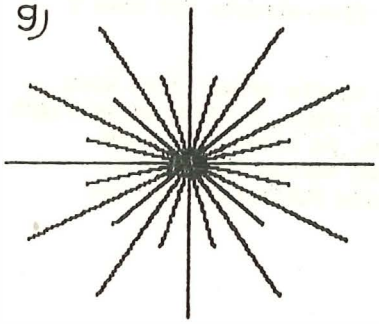
e)



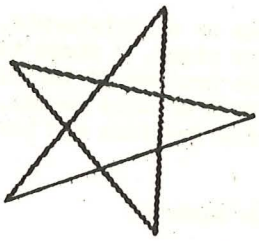
f)



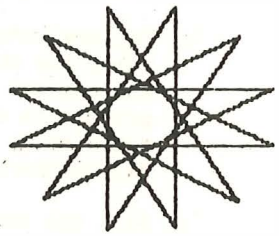
g)



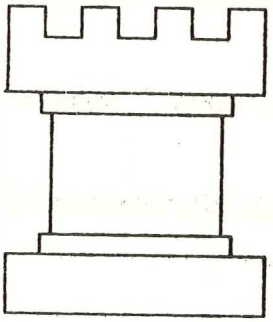
h)



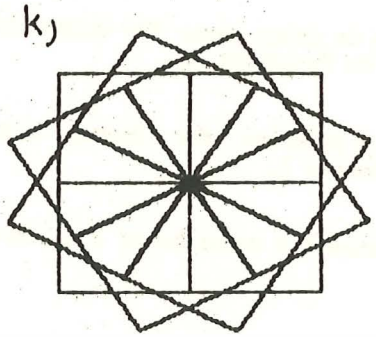
i)



j)



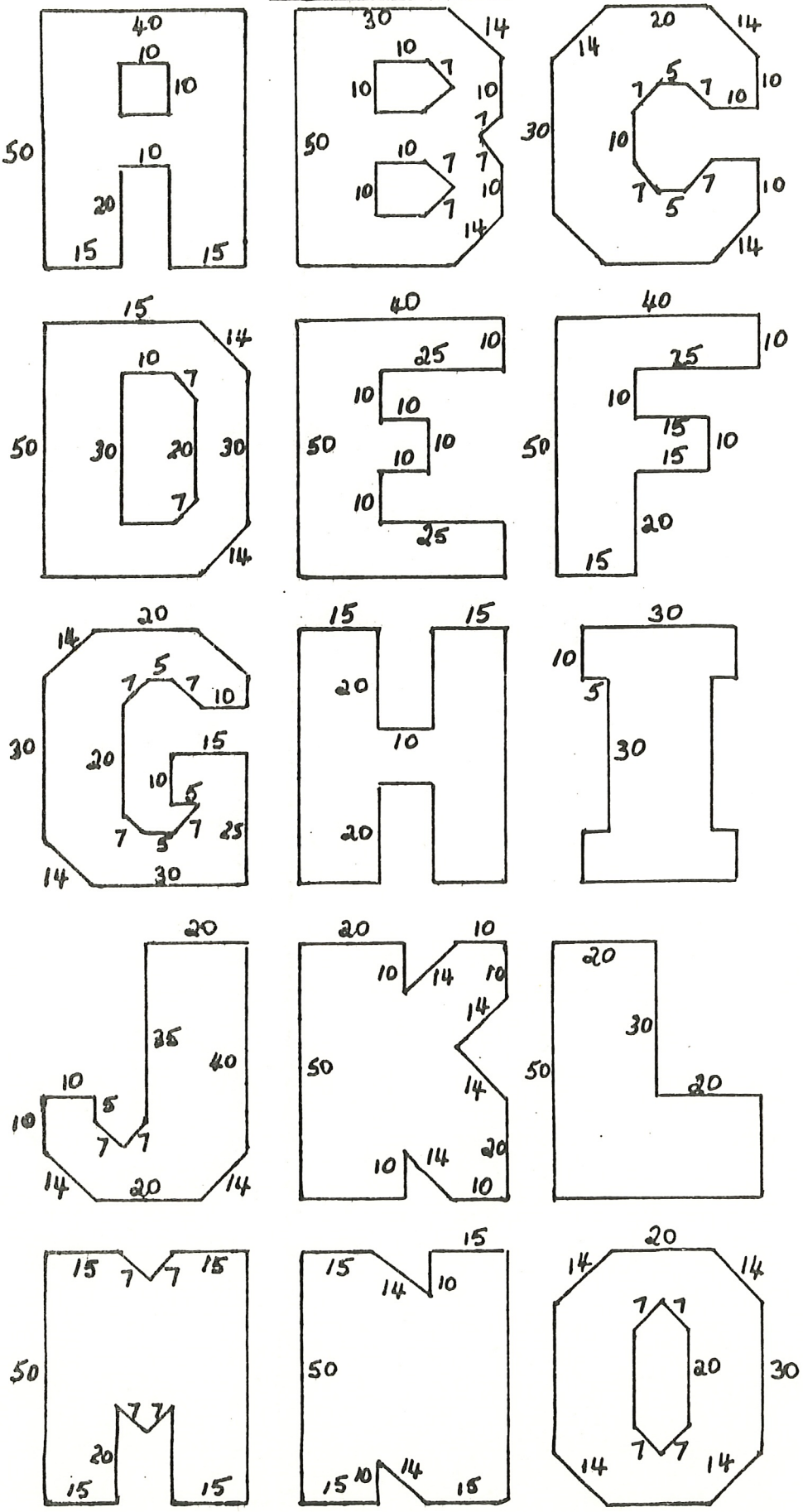
k)



APPENDIX

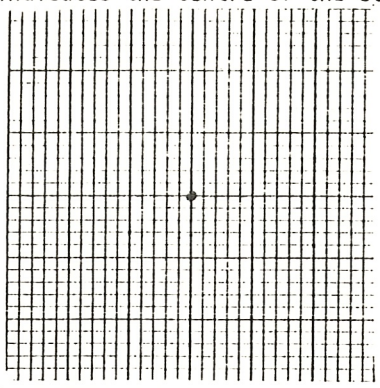
i. Worksheet 1.
ii. Worksheet 2.
iii. Worksheet 3.
iv. Worksheet 4.
v. Planning Sheet.
vi. Summary of Operation Steps.
vii. Summary of Commands and Instructions.

TURTLE WORKSHEET NO. 1

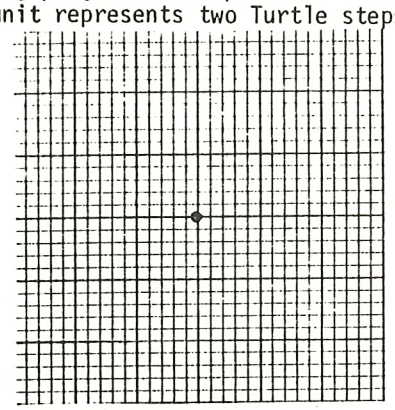


TURTLE WORKSHEET NO. 2

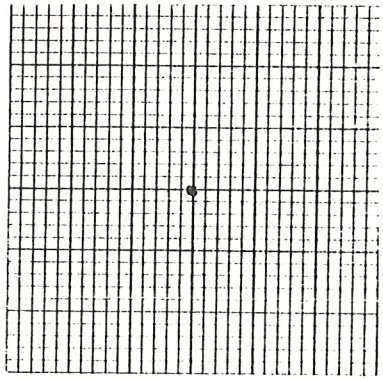
Draw on graph paper what each of the following porgrams will produce - the dot indicates the centre of the screen and one unit represents two Turtle steps.



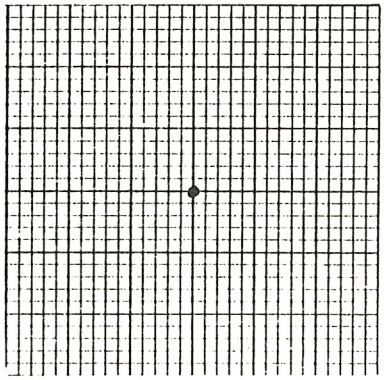
(1) F25R90F10B20



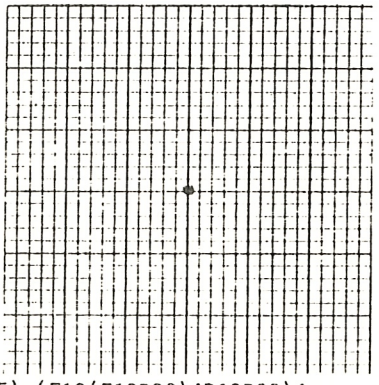
(2) UR90F10L90F5DF5(L90F20)3L90F5



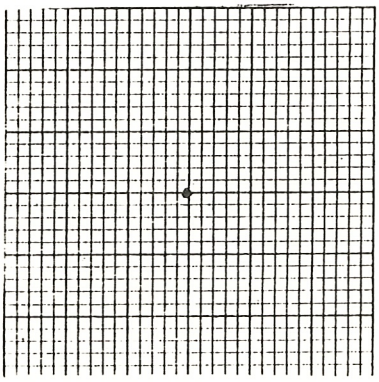
(3) UF10R90F10(R90F20)3R90F10
UR90F5DL90F5(R90F10)3R90F5



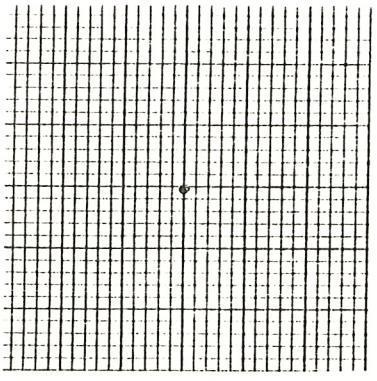
(4) UL90F20R90DF10(R45F20)7R45F10



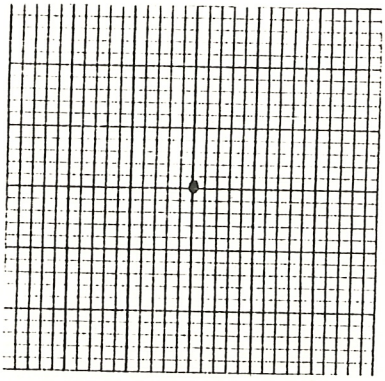
(5) (F10(F10R90)4B10R90)4



(6) UL90F15DR90F10B20F10R90F10L90F10
B20UR90F9DF2B1L90F20R90B1F2UF9
R90DF16UF2DF2



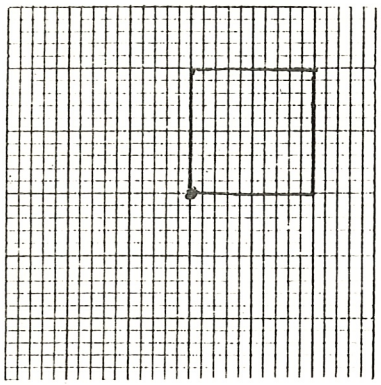
(7) R15F20R150F20R10F11



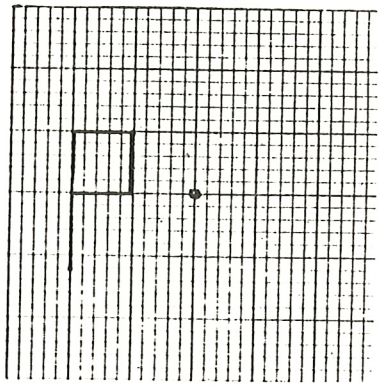
(8) UF20R90DF30R135F30L135F30

TURTLE WORKSHEET NO. 3

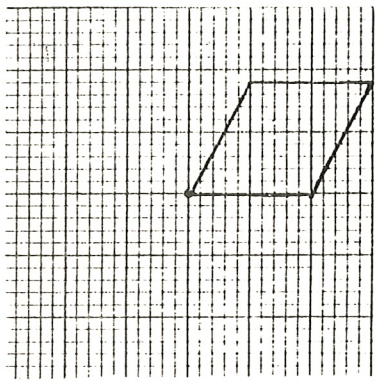
Write Turtle programs to draw the following figures - the dot indicates the centre of the screen and one unit represents two Turtle steps.



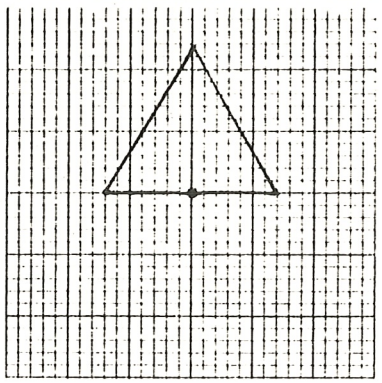
(1) _____



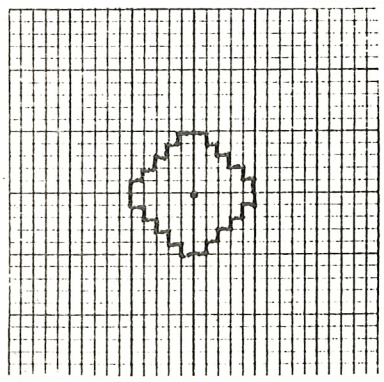
(2) _____



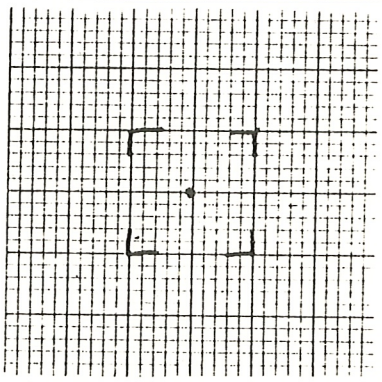
(3) _____



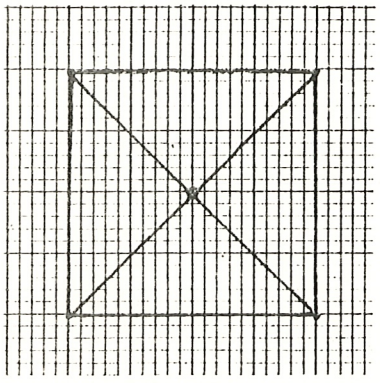
(4) _____



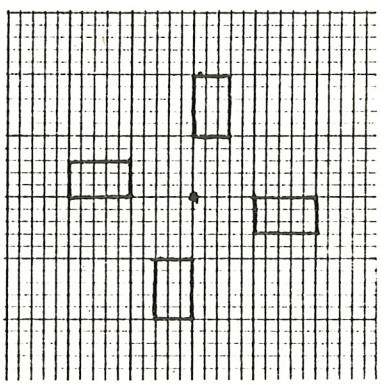
(5) _____



(6) _____



(7) _____

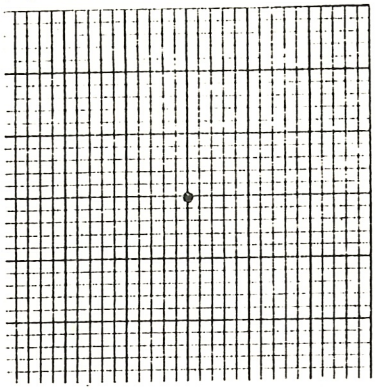


(8) _____

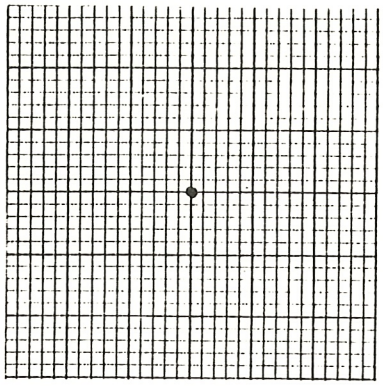
TURTLE WORKSHEET NO. 4

Draw on graph paper what each of the following programs will produce - the dot indicates the centre of the screen and one unit represents two Turtle steps.

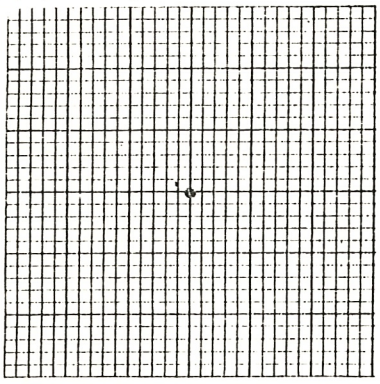
(1) F25R90F10B20



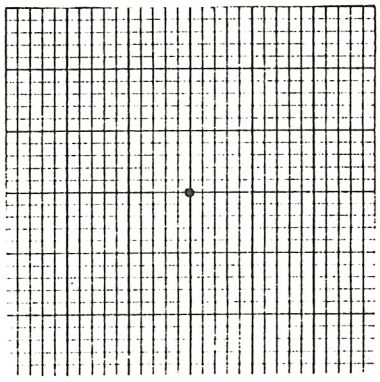
(2) B20R90F25L90F40



(3) (F25R90)4

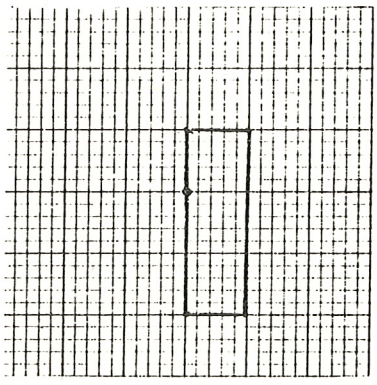


(4) F20R90UF10R90DF20

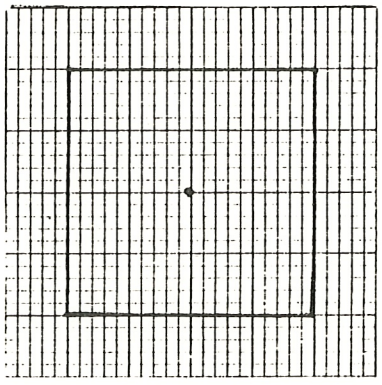


Write Turtle programs to draw the following figures - the dot indicates the centre of the screen and one unit represents two Turtle steps.

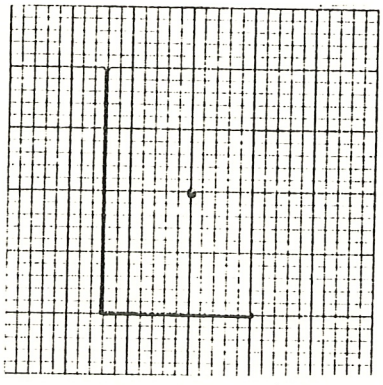
(1)



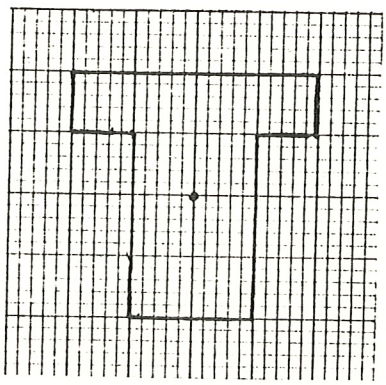
(2)

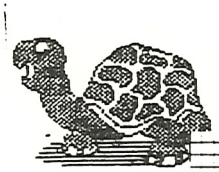


(3)



(4)





PROGRAMMABLE TURTLE PLANNING SHEET

ONE UNIT EQUALS TWO TURTLE STEPS

PROGRAMMABLE TURTLE

OPERATION STEPS

WHAT YOU SEE	WHAT YOU DO
NAME PLEASE	Type in your name (don't forget the RETURN key)
READY	C (to clear the screen)
READY	P
ENTER WHAT IS THE NAME OF YOUR PROGRAM? NAME-->	Type in the name of your program
your program name-->	Type in your program
READY	# (Hold down SHIFT and type 3)
DRAW WHAT IS YOUR PROGRAM NAME? NAME-->	Type in the name of your program
Now you may enter or edit more programs	
READY	Q to finish

****** IMPORTANT ******

PROGRAMMABLE TURTLE

INSTRUCTIONS

U	Pen Up
D	Pen Down
Fx	Forward x spaces
Bx	Back x spaces
Rx	Right x degrees
Lx	Left x degrees
(***)x	Do steps in parentheses x times
#---#	Do program named

COMMANDS

C	Clear the screen turtle home facing north pen down
P	Enter a program
#	Do a program
E	Edit a program
S	Shape (program) list
T	Turtle on/off
Q	Quit
<ESC>	Stop turtle
/	Disregard this line
--> (forward arrow)	Move cursor forward one
<-- (back arrow)	Move cursor back one
ctrl-I	Insert
ctrl-D	Delete

D. West, Government Printer, New South Wales—1982

Programmable Turtle

Vs. 2.5

COMPUTER EDUCATION UNIT

NSW Department of Education

NSW Department of Education